JMeter操作文档

## JMeter介绍

JMeter是Apache组织开发的基于Java的压力测试工具。用于对软件做压力测试，它最初被设计用于Web应用测试但后来扩展到其他测试领域。 它可以用于测试静态和动态资源例如静态文件、Java小服务程序、CGI脚本、Java 对象、数据库， FTP服务器, 等等。JMeter 可以用于对服务器、网络或对象模拟巨大的负载，来在不同压力类别下测试它们的强度和分析整体性能。另外，JMeter能够对应用程序做功能/回归测试，通过创建带有断言的脚本来验证你的程序返回了你期望的结果。为了最大限度的灵活性，JMeter允许使用正则表达式创建断言。

## JMeter运行流程

JMeter为跨平台软件，在windown和linux上都可以安装运行，JMeter是用java程序编写的，所以必须基于JDK才能运行。

一般，我们先在win上安装JMeter，通过图形化操作生成对应的.jmx配置文件，配置文件包括线程组ThreadGroup、采样器sampler、Http请求参数、相应断言配置，以及其他相关信息。

然后在linux系统上安装JMeter，运行配置好的.jmx配置文件，生成对应的测试结果。

再用win上的JMeter来查看测试结果。

接下来我们就根据以上步骤来举例说明。

### JMeter测试环境配置

前提是win和linux都已安装好了jdk，jdk请自行安装。

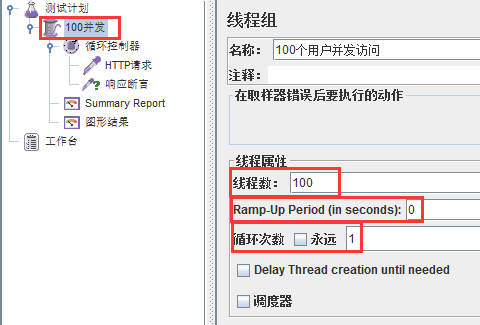
1、在win上安装并测试

访问JMeter官网下载安装包：http://jmeter.apache.org/



下载完成后，解压该安装包，进入bin目录，点击运行jmeter.bat，即可打开JMeter图形界面。

a、在测试计划中添加一个线程组

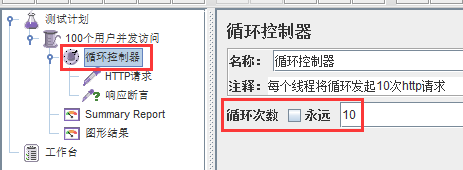


线程数为100，等同100个用户操作

Ramp-Up Period (in seconds)为0，表示100个线程需要在多久内启动完毕，0表示启动所有的线程

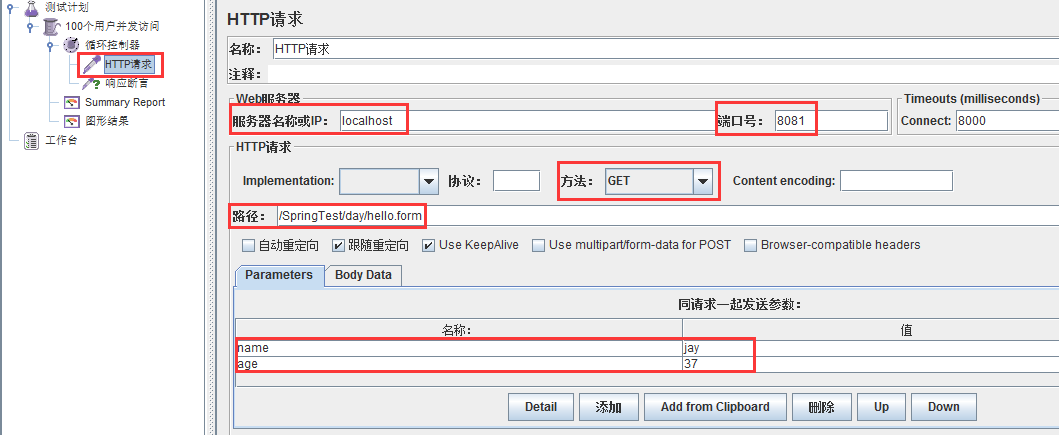
循环次数1，表示100个线程都执行一次

b、添加一个循环控制器（非必选）



选择线程组，右键选择“添加”🡪“逻辑控制器”🡪“循环控制器”，循环次数为10表示将要创建的Http请求需要在一个线程下访问10次，整个测试执行完毕后Http请求需要执行1000次

c、添加Http请求



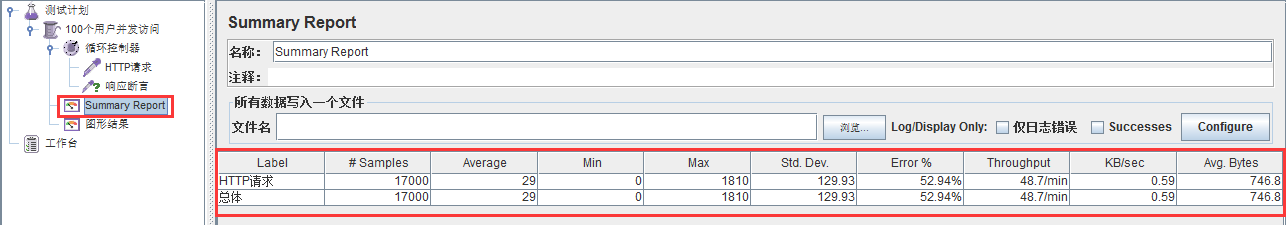
选择“循环控制器”，右键“添加”🡪“Sampler”🡪“Http请求”，在输入框中输入相应的信息，包括将要请求的ip、端口、响应时间、请求方法、请求路径、请求参数等等，我们现在以访问localhost:8081/SpringTest/day/hello.form为例测试，携带参数为name=Jay&age=37

d、添加一个响应断言



选择“循环控制器”，右键“添加”🡪“断言”🡪“响应断言”，根据响应断言来判断服务器返回的结果是否为我们预期的结果，从而判断该请求是否成功，请求的成功与否在测试结果中体现

e、添加Summary Report



选择线程组，右键选择“添加”🡪“监听器”🡪“Summary Report”，Summary Report即为“总结报告”的意思，报表中会体现测试的相关数据，其中，

Label：说明是请求类型，如Http，FTP等请求。

#Samples：也就是图形报表中的样本数目，总共发送到服务器的样本数目。

Average：也就是图形报表中的平均值，是总运行时间除以发送到服务器的请求数。

Median：也就是图形报表中的中间值，是代表时间的数字，有一半的服务器响应时间低于该值而另一半高于该值。

90%line：是指90%请求的响应时间比所得数值还要小。

Min：是代表时间的数字,是服务器响应的最短时间。

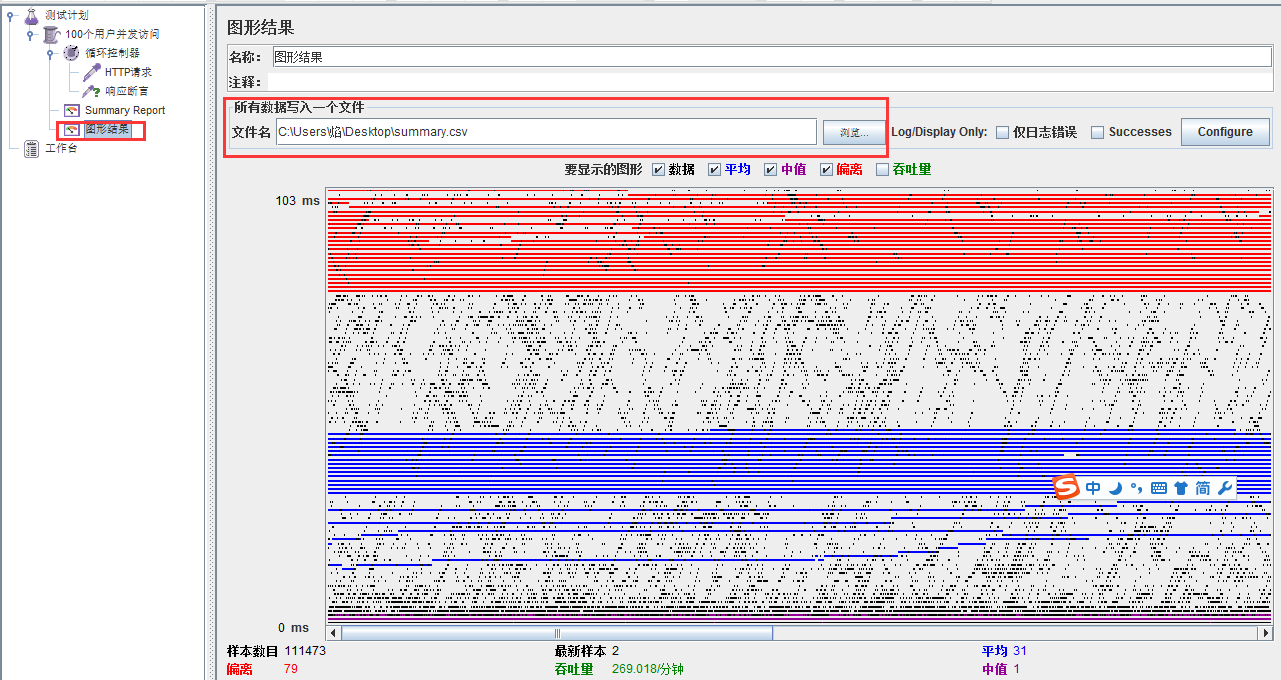
Max: 是代表时间的数字,是服务器响应的最长时间。

Error%:请求的错误百分比。

Throughput:也就是图形报表中的吞吐量，这里是服务器每单位时间处理的请求数，注意查看是秒或是分钟。

KB/sec:是每秒钟请求的字节数。

f、通过图形查看测试结果



选择线程组，右键选择“添加”🡪“监听器”🡪“图形结果”，即可图形展示测试的结果

2、在linux上安装并测试

a、登录linux

b、检查是否安装了JDK，命令

java –version

c、下载安装最新版本JMeter

wget <http://mirrors.hust.edu.cn/apache//jmeter/binaries/apache-jmeter-2.13.tgz>

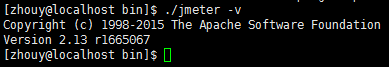
解压到当前文件夹

tar -zxvf apache-jmeter-2.13.tgz

检查JMeter版本，说明JMeter可运行

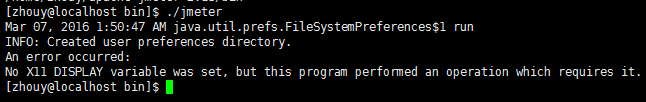
cd /home/zhouy/apache-jmeter-2.13/bin

./jmeter -v



运行jmeter

./jmeter



以上结果表示JMeter安装成功

d、设置JMeter环境变量：

切换到root用户

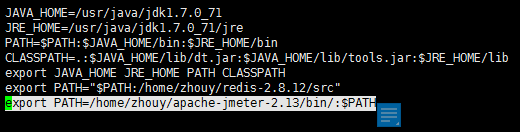
su root

输入密码

vi etc/profile

在最后面加入：

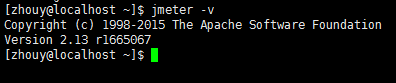
export PATH=/home/zhouy/apache-jmeter-2.13/bin/:$PATH



注销当前用户，logout后再次登录

任意位置输入

jmeter –v

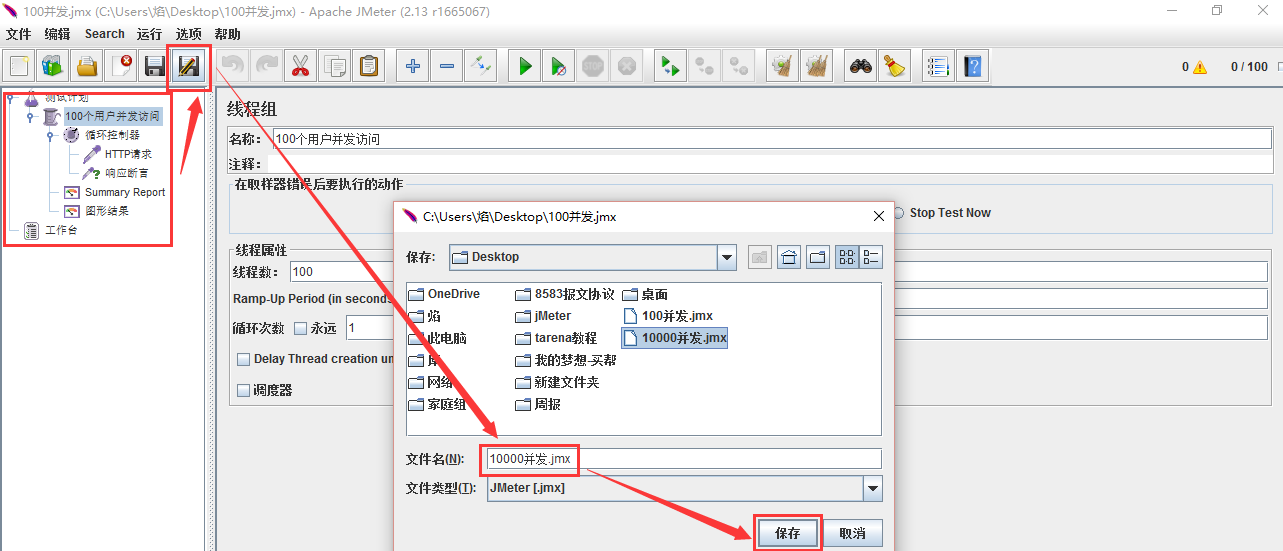


以上结果表示JMeter环境变量配置成功

e、在linux上进行测试

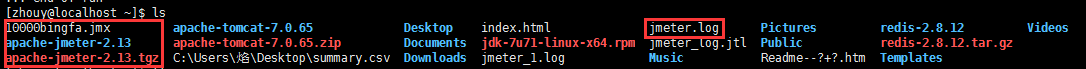
在win上准备配置文件“10000bingfa.jmx”（名字自定义）

在win上配置好数据后，选择图形界面上的“另存为”按钮，将配置文件另存为“10000bingfa.jmx”



SSH工具上传配置文件10000bingfa.jmx到服务器

JMeter相关linux的目录结构为



运行命令，注意10000bingfa.jmx和jmeter\_log.jtl的路径

jmeter -n -t 10000bingfa.jmx -l jmeter\_log.jtl

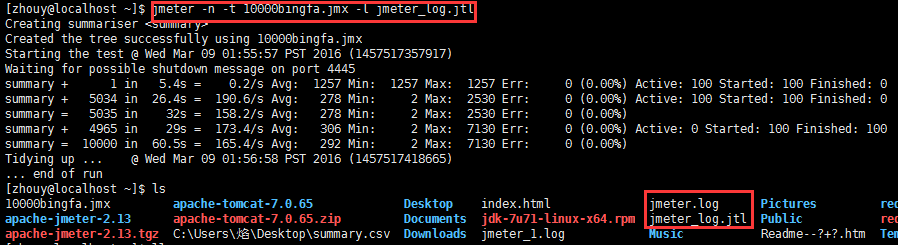
-n告知JMeter采用非图形化运行JMeter脚本

-t指定要运行的脚本文件

-l指定保存执行结果的文件名

执行命令前要检查当前目录是否是%JMeter\_Home%\bin 目录；

如果 JMeter 脚本不在当前目录，需要指定完整的路径；如果要把执行的结果保存在其他地方也要指定完整的路径。



上述结果表示jmeter测试成功，生成的log日志和jtl结果文件如上所示

这些只是最后的结果，如果我们想要及时看到统计结果呢，如下设置即可

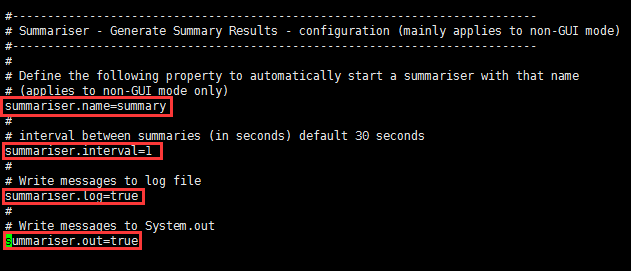
修改JMeter安装目录下bin目录下的jmeter.properties文件，取消掉对应的注释

summariser.name=summary//聚合命名

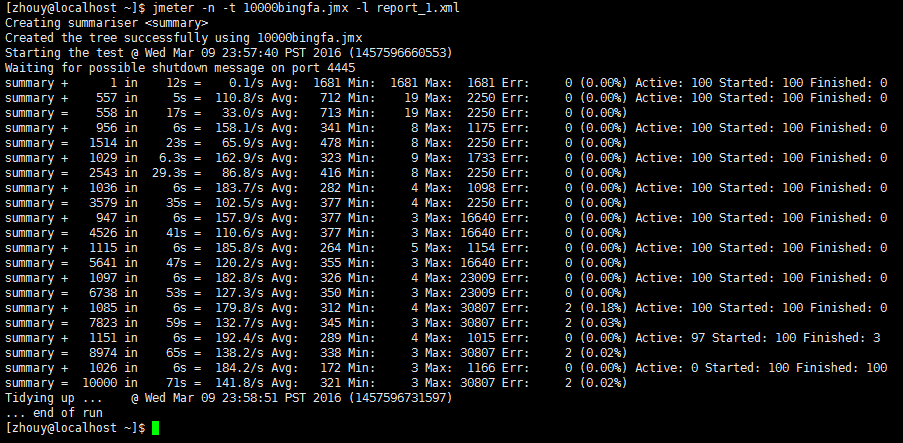
summariser.interval=1//集合刷新页面频率

summariser.log=true//是否开启日志

summariser.out=true//把信息传输至stdout



运行JMeter，实时显示数据如下

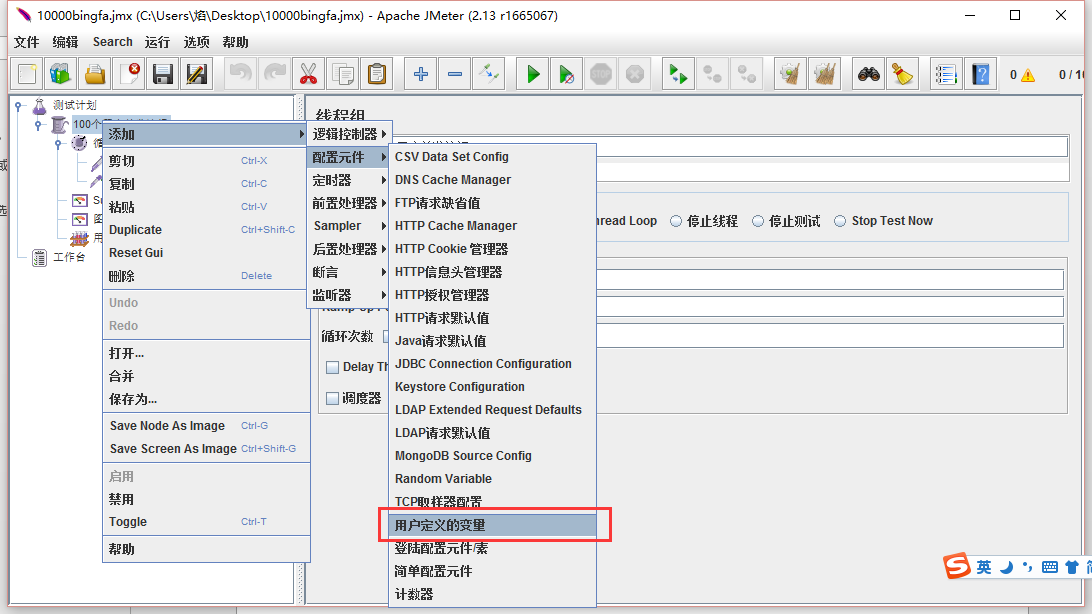


指标请参考前面讲述的图形化界面参数。

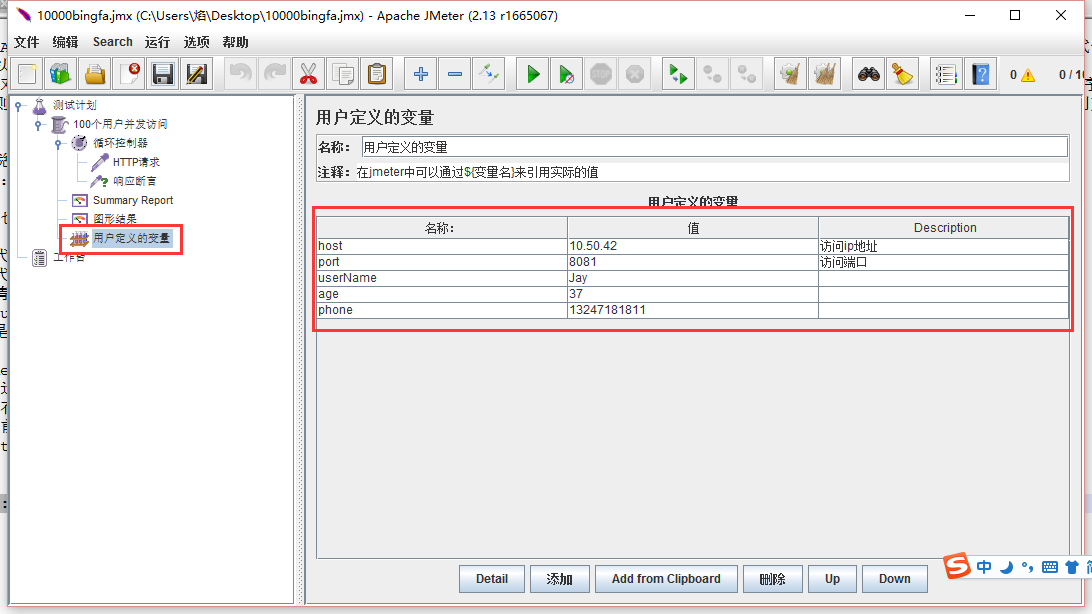
## JMeter其他相关：

### 1、用户自定义变量

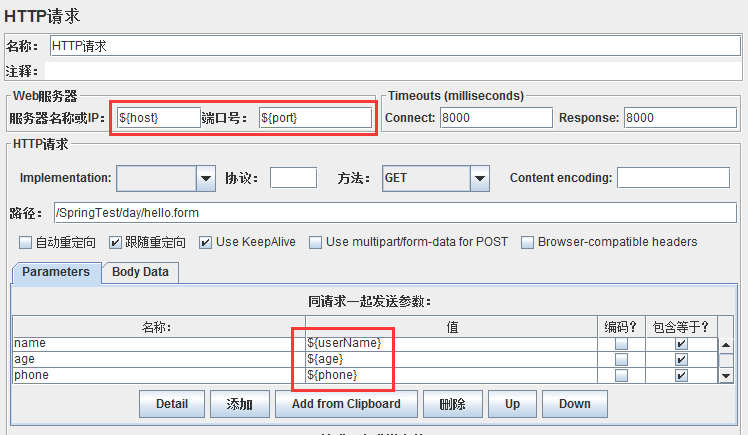
可以在JMeter中通过${变量名}来引用实际值，方便变量的管理



定义变量如下：



使用变量如下：

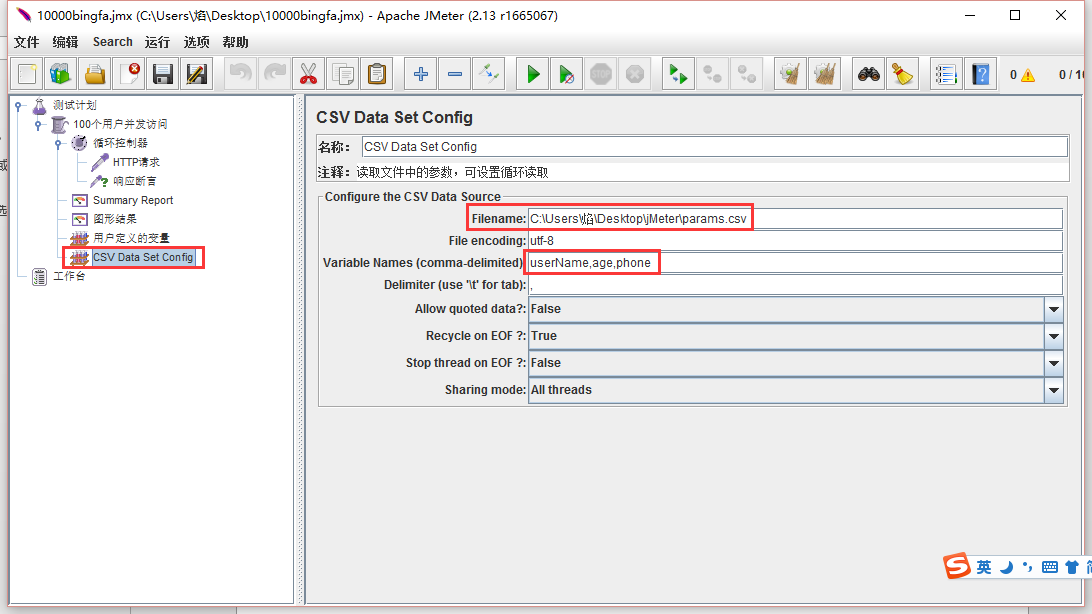


### 2、使用csv文件中的参数

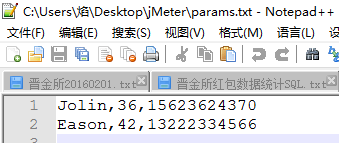
通过分隔符从文件中以一定规则读取参数值，应用场景为批量数据参数



配置如下



params.csv文件内容如下



可以先用文本编辑器编辑该文件，然后再将文件名改为.csv后缀即可。注意文件中的一行与JMeter中的Variable Names和Delimiter对应，此处对应的值分别为：

在JMeter中的使用



JMeter会根据文件中行来循环读取数据。

也可以在“用户自定义变量”中引用.csv中的变量。

### 3、在Eclipse中通过java来压力测试

流程：

1. 提供待测试的代码，或者jar包
2. 将JMeter目录apache-jmeter-2.13\lib\ext下的对应jar引入：ApacheJMeter\_core.jar、ApacheJMeter\_java.jar，添加到Eclipse中个Build Path
3. 自定义测试类和测试方法，加入相应的测试指标或者逻辑，调用待测试的jar包
4. 然后再将测试类封装成jar（也可以不封装，这里是为了协同测试的方便）
5. 在测试类中调用刚封装的jar进行压力测试

详细如下：

1、被测方法模拟

我们模拟一个被测类，在这里没有指明package，自己可以随意设定。

public class TargetClass {

private SimpleDateFormat ymdhms = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");

public void timeConsumption(){

for (int i = 0; i < 1000; i++) {

Calendar calendar = Calendar.getInstance();

Date date = calendar.getTime();

ymdhms.format(date);

}

}

public int targetFunction(String str) {

if (str != null) {

return str.length();

} else {

return -1;

}

}

}

我们看到这个类一共有两个方法，随后我们会使用JMeter对这两个方法进行一个事务处理的性能测试。targetFunction方法使我们测试的目标方法，这个方法就是返回了一个字符串的长度，timeConsumption是一个消耗时间的方法，由于目标方法耗时太小，JMeter无法统计小于1ms的方法执行时间，我们编写一个1000次的循环好耗一点时间让JMeter能做统计。将其打成jar包，eclipse里可以使用export jar file进行jar包的封装，而在实际项目使用中，可以让开发提供jar包及需要被测试的类及方法。

2、脚本代码准备

在eclipse里新建一个java工程，在 Build Path中引入JMeter目录下/lib/ext/文件夹下的ApacheJMeter\_core.jar和ApacheJMeter\_java.jar两个包，这两个包是编写性能测试脚本代码的最小包（如果需要调试运行，建议将JMeter目录下lib里所有包都引入到工程中来），然后还需要引入我们测试的包，即步骤1打的包引入到我们的脚本工程中来。

新建一个测试类，该类必须继承AbstractJavaSamplerClient虚类，在继承该类后，必须实现runTest方法，也可以根据需要去覆盖其他方法，例子的脚本代码如下(脚本内容请参考注释)：

public class TestLength extends AbstractJavaSamplerClient {

Random random = new Random();

TargetClass targetClass = new TargetClass();

//初始化方法，每个线程仅执行一次，在测试方法运行前执行

public void setupTest(JavaSamplerContext arg0) {

System.out.println("Setup");

}

//设置传入的参数，可以设置多个，已设置的参数会显示到Jmeter的参数列表中

public Arguments getDefaultParameters() {

Arguments params = new Arguments();

//定义一个参数，显示到Jmeter的参数列表中，第一个参数为参数默认的显示名称，第二个参数为默认值

params.addArgument("testStr", "");

return params;

}

//测试执行的循环体，根据线程数和循环次数的不同可执行多次，只是最主要的脚本代码，就如同http请求里的发送请求的关键脚本，返回的SampleResult作为一次请求的结果

public SampleResult runTest(JavaSamplerContext arg0) {

SampleResult results; //每一次运行的结果记录

String testStr;

results = new SampleResult();

testStr = arg0.getParameter("testStr", "");//还记得上面设置的参数吗，就和我们发送http请求时一同发送的参数一样，脚本代码中获取后可以自己来处理

if (testStr != null && testStr.length() > 0) {

results.setSamplerData(testStr);

}

try {

results.sampleStart();

//事务代码，在这里编写真实项目中远程调用的代码，sampleStart和sampleEnd方法之间作为一次事务。Jmeter把下面代码的执行时间作为一次请求的响应时间

targetClass.timeConsumption();

int result = targetClass.targetFunction(testStr);

results.sampleEnd();

//用于设置运行结果的成功或失败，如果是"false"则表示结果失败，否则则表示成功

if (result == testStr.length()) {

results.setSuccessful(true);

} else {

results.setSuccessful(false);

}

} catch (Throwable t) {

t.printStackTrace();

System.err.println("Exception!");

results.setSuccessful(false);

}

return results;

}

//结束方法，实际运行时每个线程仅执行一次，在测试方法运行结束后执行，类似于LoadRunner中的end方法

public void teardownTest(JavaSamplerContext arg0) {

System.out.println("tearDown");

}

}

红色部分是主要的事务部分，JMeter会在sampleStart时加一个时间戳，然后在sampleEnd时再加一个时间戳，两个时间戳相减JMeter则认为是我们一次Java请求的响应时间，单位ms。随后我们做一些判断可以判定此次请求成功与否，本例中应该不会有失败场景。

完成这段脚本代码后，同步骤1将项目进行jar打包，完成打包后，把本次打包jar和步骤1中的jar包放入JMeter目录的/lib/ext/目录下，然后再启动JMeter。

PS：在完成脚本的编写后，我们可以写如下一个简单的main函数对脚本进行调试，因为模拟了JMeter去执行脚本，可能需要引入更多的包到项目中来。

public static void main(String[] args) {

TestLength test = new TestLength();

JavaSamplerContext arg0 = new JavaSamplerContext(test.getDefaultParameters());

test.setupTest(arg0);

test.runTest(arg0);

test.teardownTest(arg0);

}

3、编写jmx测试计划

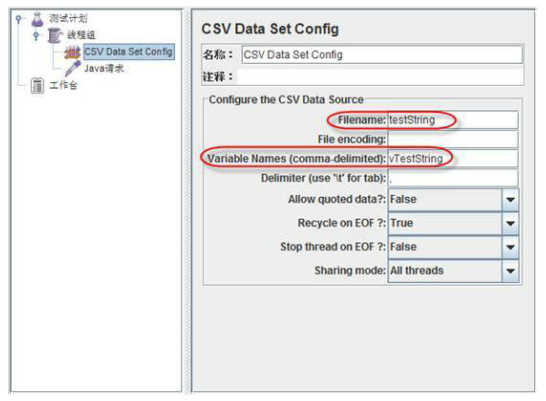
在将两次的jar放入JMeter的/lib/ext/下面后，启动JMeter，新建一个线程组，我们设置运行脚本就加载到20个线程，每个线程循环1000次，即我们对我们例子的事务进行2万次的施压。

在线程组里添加一个Java请求的Sampler，如果你已正确将脚本的jar包放入了/lib/ext/目录下，我们可以在类名称下拉框里找到刚才我们编写的脚本类com.alibaba.jmetertest.TestLength

在选中这个类后，我们还可以看到在请求里可以设置脚本里编写的入参testStr，这个值我们可以通过CSV Data Set Config、函数助手、正则表达式提取器从前一请求返回值获取等设置成一个动态的值，本例中我们使用简单的CSV Data Set Config来设置这个值吧。先新建一个CSV Data Set Config，并保存测试计划到java.jmx中。

生成1万条的随机字符串到testString文件中，将testString放到与测试计划同一目录中。

设置CSV Data Set Config内容如下



再到Java请求中我们设置我们变量的参数如下：



保存测试计划，到此我们的java.jmx已经完成，接下来我们可以执行测试计划来进行性能测试了。

4、执行测试计划

将我们生成的jar包上传到linux服务器Jmeter的/lib/ext/目录下，上传测试计划通过执行./jmeter -n -t java.jmx -l java.jtl运行脚本（如果对这条命令有疑问，可以执行jmeter.bat -h查看帮助），-n -t 后跟jmx文件，-l 指定日志文件名称。执行完毕后会在测试计划的目录下生成java.jtl文件。我们用jtlParser解析java.jtl，可以得到如下的内容：

这就是我们之前编写脚本里的事务进行2万次施压的结果，平均响应时间为47.26毫秒，事务的平均TPS为332.67。

### 4、JMeter中进行分布式测试

作为一个纯 JAVA 的GUI应用，JMeter对于CPU和内存的消耗还是很惊人的，所以当需要模拟数以千计的并发用户时，使用单台机器模拟所有的并发用户就有些力不从心， 甚至还会引起JAVA内存溢出的错误。不过，JMeter也可以像 LoadRunner 一样通过使用多台机器运行所谓的代理来分担负载产生器自身的压力，并借此来获取更大的并发用户数，我们只需手动配置一下即可。

1、在所有期望运行 JMeter作为负载产生器的机器上安装JMeter，并确定其中一台机器作为控制器，其他的机器作为代理。然后运行所有代理机器上的JMeter-server.bat文件，假定我们使用两台机器172.20.80.47和172.20.80.68作为代理；

2、在Controller机器的JMeter安装目录下找到bin 目录，再找到jmeter.properties 这个文件，使用记事本或者其他文字编辑工具打开它；

3、 在打开的文件中查找“remote\_hosts=”这个字符串，你可以找到这样一行“remote\_hosts=127.0.0.1”。其中的 127.0..0.1 表示运行JMeter代理的机器，这里需要修改为“remote\_hosts=172.20.80.47，172.20.80.68”；

4、保存文件，并重新启动控制器机器上的JMeter，并进入启动->远程启动菜单项。就会看到我们刚才添加的两个代理的地址，选中即可运行，如果想同时启动所有代理，选择远程全部启动即可。

要进行分布式测试代理机器上需要添加环境变量，即添加用户变量JMETER\_HOME＝d:\jmeter，系统变量中的path中添加d:\jmeter\bin（假设jmeter放在d盘根目录下）。